Universidad Autónoma de Tamaulipas

Unidad Académica Multidisciplinaria Mante
Universidad Autónoma de Tamaulipas

Prácticas de Laboratorio

Materia: Programación de Puertos e Interfaces

Ramón Meléndez Fabián

Luis Mario Rivera González

Juan de Dios Zapata Echeverria

05/25

Lectura de Sensor y Control de LED desde Puerto Serie en C (Linux)

Objetivo: Aprender a leer datos de un sensor desde el puerto serie en Linux, procesarlos y controlar un LED conectado a un Arduino o similar, utilizando el lenguaje C.

Materiales:

- Una computadora con Linux.
- Arduino o un microcontrolador similar que envíe datos a través del puerto serie.
- Un sensor conectado al Arduino (por ejemplo, un sensor de temperatura).
- Un LED conectado al Arduino.
- Cables de conexión.

Esquema de Conexión:

1. Conecta el Arduino al puerto USB de tu computadora.
2. Conecta el sensor a un pin analógico del Arduino.
3. Conecta el LED a un pin digital del Arduino.

Código de Arduino (Ejemplo):

```
const int ledPin = 13; // Pin del LED
int ledState = LOW; // Estado del LED
unsigned long previousMillis = 0; // Almacena el tiempo anterior
const long interval = 100; // Intervalo de tiempo en milisegundos

void setup() {
 Serial.begin(9600); // Inicializa la comunicación serial
 pinMode(ledPin, OUTPUT); // Configura el pin del LED como salida
}

void loop() {
 unsigned long currentMillis = millis(); // Obtiene el tiempo actual

 // Verifica si ha pasado el intervalo de tiempo
 if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis; // Actualiza el tiempo anterior

  int sensorValue = analogRead(A0); // Lee el valor del sensor
  Serial.println(sensorValue); // Envía el valor por el puerto serie
 }

 // Verifica si hay datos disponibles en el puerto serie
 if (Serial.available() > 0) {
  char command = Serial.read(); // Lee el comando recibido
  if (command == '1') {
   ledState = HIGH; // Enciende el LED
  } else if (command == '0') {
   ledState = LOW; // Apaga el LED
  }
  digitalWrite(ledPin, ledState); // Actualiza el estado del LED
 }
}
```

Código en C (Linux):

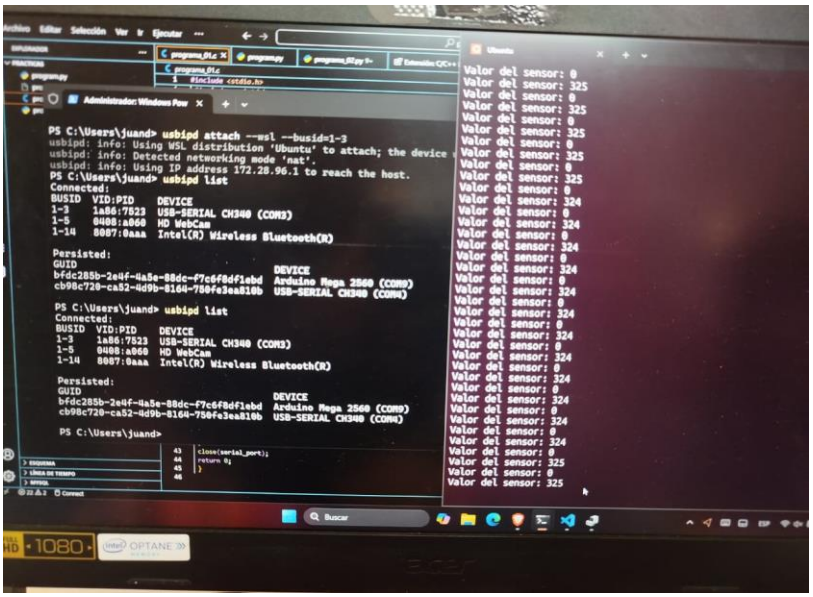
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>

int main() {
 int serial_port = open("/dev/ttyACM0", O_RDWR); // Abre el puerto serie
 if (serial_port < 0) {
  perror("Error al abrir el puerto serie");
  return 1;
 }

 struct termios tty;
 memset(&tty, 0, sizeof(tty));
 if (tcgetattr(serial_port, &tty) != 0) {
  perror("Error al obtener atributos del puerto serie");
  return 1;
 }

 tty.c_cflag = CS8 | CREAD | CLOCAL; // Configura el puerto serie
 tty.c_cc[VMIN] = 1; // Lee al menos 1 carácter
 tty.c_cc[VTIME] = 5; // Espera hasta 0.5 segundos para la entrada

 if (tcsetattr(serial_port, TCSANOW, &tty) != 0) {
  perror("Error al establecer atributos del puerto serie");
  return 1;
 }

 char buffer[256];
 int bytes_read;
 int sensorValue;

 while (1) {
  bytes_read = read(serial_port, buffer, sizeof(buffer));
  if (bytes_read > 0) {
   buffer[bytes_read] = '\0'; // Termina la cadena
   sensorValue = atoi(buffer); // Convierte la cadena a entero
   printf("Valor del sensor: %d\n", sensorValue);

   if (sensorValue > 500) { // Umbral para encender el LED
    write(serial_port, "1\n", 2); // Envía "1" para encender el LED
   } else {
    write(serial_port, "0\n", 2); // Envía "0" para apagar el LED
```

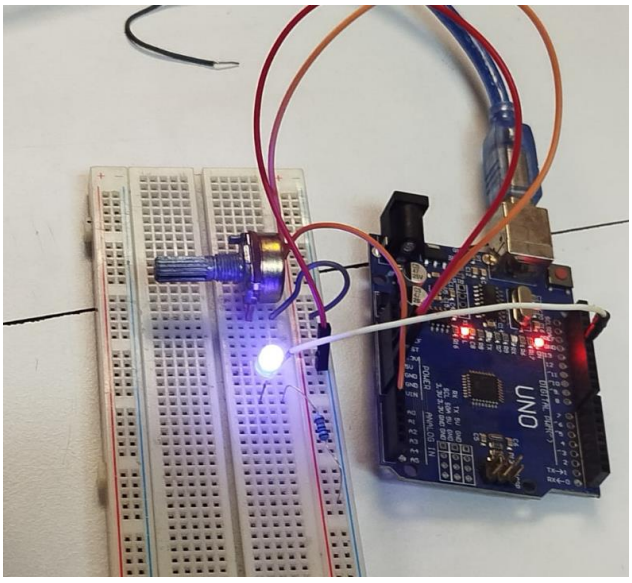
Practica realizada:

playground-1.mongodb.js ● | programa_01.c ✕ | programa_02.py ●

C: > Users > monme > OneDrive > Desktop > C programa_01.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
int main() {
int serial_port = open("/dev/ttyUSB0", O_RDWR); // Abre el puerto serie
if (serial_port < 0) {
perror("Error al abrir el puerto serie");
return 1;
}
struct termios tty;
memset(&tty, 0, sizeof(tty));
if (tcgetattr(serial_port, &tty) != 0) {
perror("Error al obtener atributos del puerto serie");
return 1;
}
tty.c_cflag = CS8 | CREAD | CLOCAL; // Configura el puerto serie
tty.c_cc[VMIN] = 1; // Lee al menos 1 carácter
tty.c_cc[VTIME] = 5; // Espera hasta 0.5 segundos para la entrada
if (tcsetattr(serial_port, TCSANOW, &tty) != 0) {
perror("Error al establecer atributos del puerto serie");
return 1;
}
char buffer[256];
int bytes_read;
int sensorValue;
while (1) {
bytes_read = read(serial_port, buffer, sizeof(buffer));
if (bytes_read > 0) {
buffer[bytes_read] = '\0'; // Termina la cadena
sensorValue = atoi(buffer); // Convierte la cadena a entero
printf("Valor del sensor: %d\n", sensorValue);
if (sensorValue > 500) { // Umbral para encender el LED
write(serial_port, "1\n", 2); // Envía "1" para encender el LED
} else {
write(serial_port, "0\n", 2); // Envía "0" para apagar el LED
}
}
usleep(100000); // Espera 0.1 segundos
```





Lectura de Sensor y Control de LED desde Puerto Serie en Python (Linux)

Objetivo: Aprender a leer datos de un sensor desde el puerto serie en Linux, procesarlos y controlar un LED conectado a un Arduino o similar, utilizando Python.

Materiales:

- Una computadora con Linux.
- Arduino o un microcontrolador similar que envíe datos a través del puerto serie.
- Un sensor conectado al Arduino (por ejemplo, un sensor de temperatura).
- Un LED conectado al Arduino.
- Cables de conexión.

Esquema de Conexión:

1. Conecta el Arduino al puerto USB de tu computadora.
2. Conecta el sensor a un pin analógico del Arduino.
3. Conecta el LED a un pin digital del Arduino.

Código de Arduino (Ejemplo):

```
const int ledPin = 13; // Pin del LED
int ledState = LOW; // Estado del LED
unsigned long previousMillis = 0; // Almacena el tiempo anterior
const long interval = 100; // Intervalo de tiempo en milisegundos

void setup() {
 Serial.begin(9600); // Inicializa la comunicación serial
 pinMode(ledPin, OUTPUT); // Configura el pin del LED como salida
}

void loop() {
 unsigned long currentMillis = millis(); // Obtiene el tiempo actual
 // Verifica si ha pasado el intervalo de tiempo
 if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis; // Actualiza el tiempo anterior
  int sensorValue = analogRead(A0); // Lee el valor del sensor
  Serial.println(sensorValue); // Envía el valor por el puerto serie
 }

 // Verifica si hay datos disponibles en el puerto serie
 if (Serial.available() > 0) {
  char command = Serial.read(); // Lee el comando recibido
  if (command == '1') {
   ledState = HIGH; // Enciende el LED
  } else if (command == '0') {
   ledState = LOW; // Apaga el LED
  }
  digitalWrite(ledPin, ledState); // Actualiza el estado del LED
 }
}
```

Código en Python (Linux):

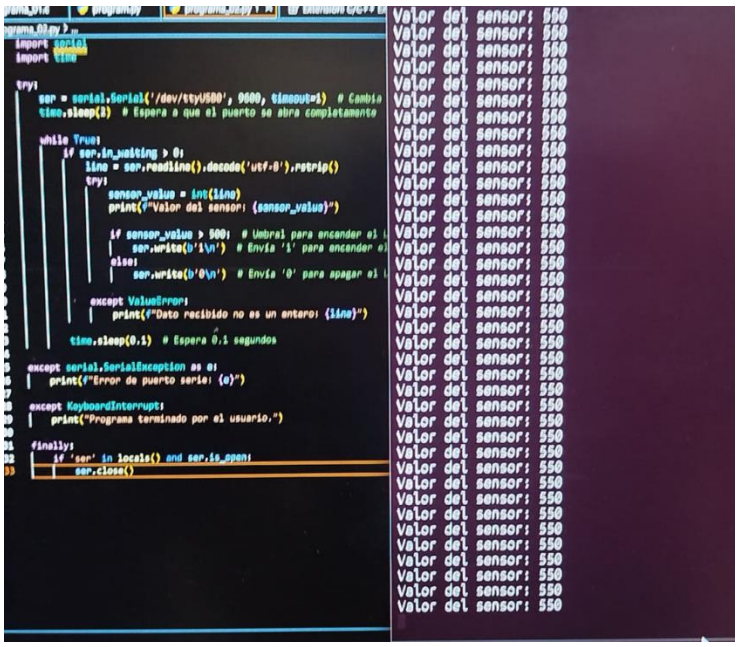
```
  import serial
  import time

  try:
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)  # Cambia '/dev/ttyACM0' si es
necesario
    time.sleep(2)  # Espera a que el puerto se abra completamente

    while True:
      if ser.in_waiting > 0:
        line = ser.readline().decode('utf-8').rstrip()
        try:
          sensor_value = int(line)
          print(f"Valor del sensor: {sensor_value}")

          if sensor_value > 500:  # Umbral para encender el LED
            ser.write(b'1\n')  # Envía '1' para encender el LED
          else:
            ser.write(b'0\n')  # Envía '0' para apagar el LED
        except ValueError:
          print(f"Dato recibido no es un entero: {line}")
      time.sleep(0.1)  # Espera 0.1 segundos

  except serial.SerialException as e:
    print(f"Error de puerto serie: {e}")
  except KeyboardInterrupt:
    print("Programa terminado por el usuario.")
  finally:
    if 'ser' in locals() and ser.is_open:
      ser.close()
    }
    digitalWrite(ledPin, ledState);
   }
  }
```

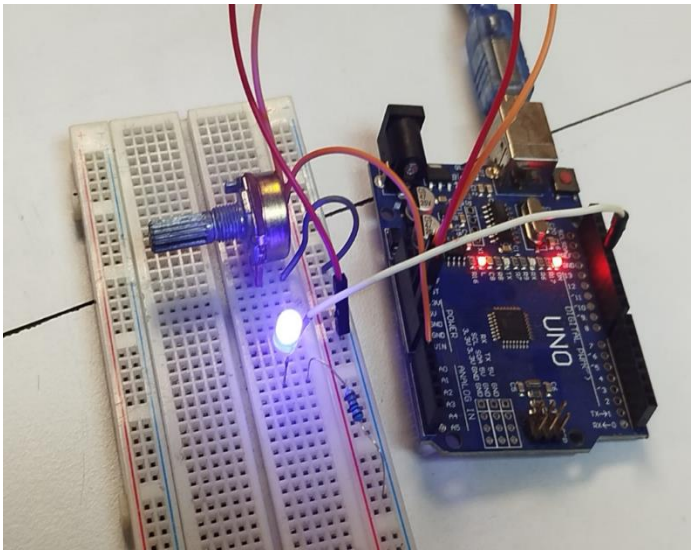
Practica Realizada:

playground-1.mongodb.js ● programa_01.c programa_02.py ●

C: > Users > monme > OneDrive > Desktop > programa_02.py

```python
#RAMÓN Meléndez Fabián
#Luis Mario Rivera Gonzales
#Juan de Dios Zapata Echeverria
import serial
import time

try:
    ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)  # Cambia '/dev/ttyACM0' si es necesario
    time.sleep(2)  # Espera a que el puerto se abra completamente

    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            try:
                sensor_value = int(line)
                print(f"Valor del sensor: {sensor_value}")

                if sensor_value > 500:  # Umbral para encender el LED
                    ser.write(b'1\n')  # Envía '1' para encender el LED
                else:
                    ser.write(b'0\n')  # Envía '0' para apagar el LED

            except ValueError:
                print(f"Dato recibido no es un entero: {line}")

        time.sleep(0.1)  # Espera 0.1 segundos

except serial.SerialException as e:
    print(f"Error de puerto serie: {e}")

except KeyboardInterrupt:
    print("Programa terminado por el usuario.")

finally:
    if 'ser' in locals() and ser.is_open:
        ser.close()
```





Lectura de Sensor y Control de LED desde Puerto Serie en Python (Windows)

Objetivo: Aprender a leer datos de un sensor desde el puerto serie en Windows, procesarlos y controlar un LED conectado a un Arduino o similar, utilizando Python.

Materiales:

- Una computadora con Windows.
- Arduino o un microcontrolador similar que envíe datos a través del puerto serie.
- Un sensor conectado al Arduino (por ejemplo, un sensor de temperatura).
- Un LED conectado al Arduino.
- Cables de conexión.

Esquema de Conexión:

1. Conecta el Arduino al puerto USB de tu computadora.
2. Conecta el sensor a un pin analógico del Arduino.
3. Conecta el LED a un pin digital del Arduino.

Código de Arduino (Ejemplo):

```
   const int ledPin = 13; // Pin del LED
int ledState = LOW; // Estado del LED
unsigned long previousMillis = 0; // Almacena el tiempo anterior
const long interval = 100; // Intervalo de tiempo en milisegundos

void setup() {
 Serial.begin(9600); // Inicializa la comunicación serial
 pinMode(ledPin, OUTPUT); // Configura el pin del LED como salida
}

void loop() {
 unsigned long currentMillis = millis(); // Obtiene el tiempo actual

 // Verifica si ha pasado el intervalo de tiempo
 if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis; // Actualiza el tiempo anterior

  int sensorValue = analogRead(A0); // Lee el valor del sensor
  Serial.println(sensorValue); // Envía el valor por el puerto serie
 }

 // Verifica si hay datos disponibles en el puerto serie
 if (Serial.available() > 0) {
  char command = Serial.read(); // Lee el comando recibido
  if (command == '1') {
   ledState = HIGH; // Enciende el LED

  } else if (command == '0') {
   ledState = LOW; // Apaga el LED
  }
  digitalWrite(ledPin, ledState); // Actualiza el estado del LED
 }
}
```

Código en Python (Windows):

Python

```
import serial
import time

try:
  ser = serial.Serial('COM3', 9600, timeout=1)  # Cambia 'COM3' si es necesario
  time.sleep(2)  # Espera a que el puerto se abra completamente

  while True:
    if ser.in_waiting > 0:
      line = ser.readline().decode('utf-8').rstrip()
      try:
        sensor_value = int(line)
        print(f"Valor del sensor: {sensor_value}")

        if sensor_value > 500:  # Umbral para encender el LED
          ser.write(b'1\n')  # Envía '1' para encender el LED
        else:
          ser.write(b'0\n')  # Envía '0' para apagar el LED
      except ValueError:
        print(f"Dato recibido no es un entero: {line}")
    time.sleep(0.1)  # Espera 0.1 segundos

except serial.SerialException as e:
  print(f"Error de puerto serie: {e}")
except KeyboardInterrupt:
  print("Programa terminado por el usuario.")
finally:
  if 'ser' in locals() and ser.is_open:
    ser.close()
```

Practica Realizada:

playground-1.mongodb.js ● | programa_01.c | programa_02.py ●

C: > Users > monme > OneDrive > Desktop > programa_02.py

```python
#RAMÓN Meléndez Fabián
#Luis Mario Rivera Gonzales
#Juan de Dios Zapata Echeverria
import serial
import time

try:
    ser = serial.Serial('COM3', 9600, timeout=1)  # Cambia '/dev/ttyACM0' si es necesario
    time.sleep(2)  # Espera a que el puerto se abra completamente

    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            try:
                sensor_value = int(line)
                print(f"Valor del sensor: {sensor_value}")

                if sensor_value > 500:  # Umbral para encender el LED
                    ser.write(b'1\n')  # Envía '1' para encender el LED
                else:
                    ser.write(b'0\n')  # Envía '0' para apagar el LED

            except ValueError:
                print(f"Dato recibido no es un entero: {line}")

        time.sleep(0.1)  # Espera 0.1 segundos

except serial.SerialException as e:
    print(f"Error de puerto serie: {e}")

except KeyboardInterrupt:
    print("Programa terminado por el usuario.")

finally:
    if 'ser' in locals() and ser.is_open:
        ser.close()
```